Rice Roadmap - Release Cycles

- Release Versioning Scheme
- Release Lifespan
  - Major Version Lifespan
  - Minor Version Lifespan
  - Patch Version Lifespan
  - Lifespan Summary
- Relationship of Kuali Rice Releases to Kuali Application Releases
- End-Of-Life for Major Versions
- Release Cycle
- Pre-Release Versions
  - Development Builds
  - Milestone Releases
  - Release Candidates

This roadmap shall outline the intended major and minor releases, along with assumed timeframes and expectations for delivering tier 1 and 2 enhancements. It will suggest tier 3 out of scope enhancements that other interested parties are encouraged to deliver within similar timeframes.

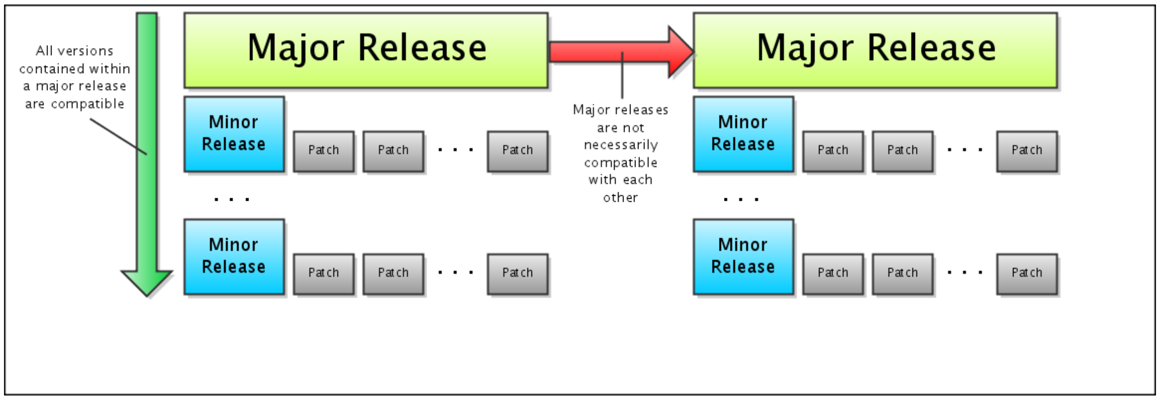
The roadmap and the release cycle are inter-dependent. In order to plan how functionality and technology are evolved in Rice, we need to understand what the release cycle will look like, and how it will relate to the releases of Kuali Applications (KFS, KS, KC, etc.).

Release Versioning Scheme

Versions of Kuali Rice shall follow a numerical scheme consisting of three parts separated by periods: Major.Minor.Patch

- **Major Versions** are intended to be long-lived versions of the software that consist of numerous minor versions. The decision to create a new major version will be the result of the need for major changes that cannot be successfully implemented without breaking version compatibility. It is permitted that different major versions of the software will not be compatible with each other. Applications that are created in later versions of Rice may not be compatible with earlier major versions of Rice. However, a reasonable and well-documented upgrade path should still be made available from a previous major version of the software to it's next major version.
- **Minor Versions** are where most of the new work on the software will be done. It is intended that different minor versions within the same major version should be compatible.
- **Patch Versions** should be created to address security issues or bugs with an existing minor version release. Different patch versions within the same minor version should be fully compatible with each other.

The following graphic illustrates the relationship between the different version types.


All versions contained within a major release are compatible
Major Release
Minor Release
Patch
Patch
Patch
. . .
Major releases are not necessarily compatible with each other
Major Release
Minor Release
Patch

Release Lifespan

Each of the different types of Kuali Rice versions will have an expected lifespan which will help to determine how best to schedule roadmap items into a release.

Major Version Lifespan

As mentioned previously, a major version of Kuali Rice should represent large architectural changes or paradigm shifts which could result in changes that cause incompatibility with previous versions of the software. Efforts should be made to reduce this impact as much as possible. However, in order to allow for continued evolution of the Kuali Rice software, these types of changes will be necessary.

Additionally, a major version will likely be started in parallel with work happening on a previous major version. Subsequently, once a new major version is released, maintenance work will continue on the previous major version until it's end-of-life (see End-Of-Life below).

Because of the impact that major versions will have on existing applications, new major versions should be created only when necessary and should have a long lifespan.

Ideally, a new major version of Kuali Rice should not occur any more frequently then every **2-3 years**.

Minor Version Lifespan

Minor versions will be where the majority of the new Kuali Rice work will happen. Generally, other Kuali applications will work to align their releases with a specific minor version of Rice. The goal for minor versions of Rice should be to maintain a relatively short release cycle in order to facilitate pushing out new functionality for Kuali applications or other implementers in a reasonable time line.

However, minor versions of the software will still need to go through a rigorous quality assurance cycle in order to ensure that the version is as stable as possible. This will require that sufficient time be allocated for this during the release cycle of the minor version.

Taking all of this into consideration, the Kuali Rice team should strive for minor version releases every **6-9 months**.

Patch Version Lifespan

Patch versions are to be primarily used for bug fixes and security issues. At a minimum, the two most recent minor versions of Kuali Rice (within a specific major version) at any given time will continue to be maintained with patches.

Because of the bug fixing and security focus of patch releases, these should be released **as often as needed**.

Lifespan Summary

Major Versions: 2-3 years
Minor Versions: 6-9 months
Patch Versions: as needed

Relationship of Kuali Rice Releases to Kuali Application Releases

As mentioned previously, Kuali Applications (KFS, KC, KS, etc.) should attempt to align their release schedule with a specific minor release of Kuali Rice. They should then work with the roadmap committees to ensure that any Kuali Rice work that they require for their release be included in the roadmap and release planning for that version of Rice.

Ideally, the Kuali Rice work will be done in advance of the application work and be completed prior to the Kuali application's quality assurance period to allow time for the Kuali application to properly integrate and test the changes. In practice, this can be difficult to accomplish. However, this should be a goal of increasing importance as we continue to move Kuali Rice and the other Kuali applications forward.

The targeted version of Kuali Rice should be released to the public prior or in conjunction with the public release of the Kuali application software on which it is dependent.

End-Of-Life for Major Versions

In order to reduce the overhead of maintaining numerous old versions of the Kuali Rice software, an end-of-life policy will be implemented for older major versions.

A major version will reach it's end-of-life once two subsequent major versions have been released.

For example, Kuali Rice 1.x will reach it's end-of-life once Kuali Rice 3.0 is released.

During the intervening period, patches will continue to be made available when needed for the previous major version of the software.

Release Cycle

The Kuali Rice release cycle consists of a few distinct phases as outlined in the charter.

1. Preparation and Planning
2. Application & Technology Architecture
3. Software Design and Development
4. Software Testing and Configuration
5. Software Release & Implementation Support
6. Post-Release Support

Note that these phases are applicable to major and minor version work. Patch versions will be driven based on bug reports from the community and other Kuali applications and will happen during the "Post-Release Support" phase.

Pre-Release Versions

In order to allow for pre-release testing and upgrade planning by the community, Kuali Rice will provide early availability releases of the software to the public. Each of these "releases" will fall into one of three categories:

1. Development Builds
2. Milestone Releases
3. Release Candidates

The implications of making these versions available early include:

1. Licensing vigilance needs to be maintained in an ongoing fashion on the project. All licensing policies and procedures must be followed rigorously in order to prevent accidental release of code or binaries which are using improperly licensed code or libraries.
2. Release notes and changelog documentation needs to be maintained and updated in parallel and in real-time to the development work. This will facilitate others being able to successfully work with the early releases and test them prior to the generally available packaged release and documentation.

Development Builds

The development process of Kuali Rice is backed by a continuous integration environment in which the software is continually built and automatically tested. This environment can be used to create development builds of the software for those that are interested in working with very early and possibly unstable versions of the software. The primary customers for these builds would be those in the community that might want to apply patches to their local implementation or the other Kuali applications which might be working with these development builds in parallel with development of their application.

Alternatively, the source code for the latest version will be publicly available at all times in source control for those that want to obtain a copy of it.

Milestone Releases

A milestone release will represent a point in time in the release and development cycle when a major set of enhancements or changes are completed. A minor version may only have one or two milestones while a new major version may have many more.

What constitutes the milestones for a particular version of Kuali Rice will be decided during the planning phase and will be based on the major roadmap items of which that release is composed. A milestone release should be partially stable such that those who want to begin working with it can. It should be accompanied by some documentation which gives interested parties the information they need to get started with it. However, it is not required that this documentation will be complete.

Generally speaking, each version should have at least one milestone release prior to the beginning of the Quality Assurance phase.

Release Candidates

A release candidate represents a version of the software with potential to be the final public release. The intention of making a release candidate available is to allow for those in the community to work with the software and report any bugs they may find. If there are no fatal bugs found with the release candidate within a specified period of time, then the version is ready for release. If fatal bugs are found, they will be fixed and a subsequent release candidate will be made available. A release candidate should be accompanied with a mostly complete set of documentation.

Ideally, each version of the software should only go through 1 or 2 release candidate versions prior to the generally available public release.